

Contents

1	Routine/Function Prologues	2
1.0.1	readgeos.F90 (Source File: readgeos.F90)	2
1.1	Core Functions of readgeos	2
1.1.1	check_error (Source File: readgeos.F90)	8

1 Routine/Function Prologues

1.0.1 readgeos.F90 (Source File: readgeos.F90)

Reads in GEOS data and performs interpolation to the LDAS domain.

1.1 Core Functions of readgeos

polates0 Interpolates GEOS data to LDAS grid using polates0 (bilinear)

GEOS FORCING VARIABLES (unless noted, fields are 3-hr upstream averaged):

1. T 2m Temperature interpolated to 2 metres [K]
2. q 2m Instantaneous specific humidity interpolated to 2 metres [kg/kg]
3. radswg Downward shortwave flux at the ground [W/m^2]
4. lwgdown Downward longwave radiation at the ground [W/m^2]
5. u 10m Instantaneous zonal wind interpolated to 10 metres [m/s]
6. v 10m Instantaneous meridional wind interpolated to 10 metres [m/s]
7. ps Instantaneous Surface Pressure [Pa]
8. preacc Total precipitation [mm/s]
9. precon Convective precipitation [mm/s]
10. albedo Surface albedo (0-1)

REVISION HISTORY:

- 1 Oct 1999: Jared Entin; Initial code
- 15 Oct 1999: Paul Houser; Significant F90 Revision
- 11 Apr 2000: Brian Cosgrove; Added read statements for forcing interpolation
- 17 Apr 2001: Jon Gottschalck; Added code to perform initialization of
Mosaic with GEOS forcing and new intp. scheme
- 14 Aug 2001: Urszula Jambor; Added ferror flag as a routine argument
- 07 Dec 2001: Urszula Jambor; Began used of LDAS%\$LDAS_KGDS array

INTERFACE:

```
subroutine readgeos(order,name,ld,tscount,ferror)
```

USES:

```

use lis_module      ! LDAS non-model-specific 1-D variables
use spmdMod
use baseforcing_module, only: glbdata1, glbdata2
#if ( defined BUDGETIP )
  use def_ipMod, only : w110,w120,w210,w220,n110,n120,n210,n220,rlat0,rlon0,&
                        w113,w123,w213,w223,n113,n123,n213,n223,rlat3,rlon3,mo
#else
  use def_ipMod, only : w110,w120,w210,w220,n110,n120,n210,n220,rlat0,rlon0,mo
#endif
use geosdomain_module, only : geosdrv,mi

#if ( defined OPENDAP )
```

```

    use tile_spmMod
    use geosopendap_module, only : geos_slat, geos_nlat, &
        cgeos_slat, cgeos_nlat, geos_nr, geos_nc, &
        get_geos_index
    use opendap_module, only : ciam, parm_nc, parm_nr, &
        nroffset
#endif
    use lisdrv_module, only : gindex ! LDAS non-model-specific 1-D variables

```

CONTENTS:

```

#if ( defined OPENDAP )
    integer :: nr_index, nc_index
    integer :: nr_dom, nc_dom
    integer :: geos_index
    character*4 :: cgeos_index
    real, allocatable :: debuggeos0(:, :)
    real, allocatable :: debuggeos1(:, :)
    integer :: ngrid1
    integer :: nmif1
    integer :: ierr, status(MPI_STATUS_SIZE)
    nr_index = geos_nr
    nc_index = geos_nc
    nr_dom = parm_nr!ld%lnr
    nc_dom = parm_nc!ld%lnc
#else
    integer :: nr_index, nc_index
    integer :: nr_dom, nc_dom
    integer :: fnroffset=0
    integer :: nroffset=0
    integer :: geos_nc=0
    nr_index = geosdrv%nrold
    nc_index = geosdrv%ncold
    nr_dom = ld%lnr
    nc_dom = ld%lnc
#endif

    allocate(tempvar(nc_index, nr_index, geosdrv%nmif), stat=ios)
    call check_error(ios, 'Error allocating tempvar.', iam)

    allocate(tempgeos(ld%ngrid, geosdrv%nmif), stat=ios)
    call check_error(ios, 'Error allocating tempgeos.', iam)

    allocate(f(nc_index*nr_index), stat=ios)
    call check_error(ios, 'Error allocating f.', iam)

    allocate(go(nc_dom*nr_dom), stat=ios)
    call check_error(ios, 'Error allocating go.', iam)

```

```

allocate(gmask(nc_index*nr_index), stat=ios)
call check_error(ios,'Error allocating gmask.',iam)

allocate(lb(nc_index*nr_index), stat=ios)
call check_error(ios,'Error allocating lb.',iam)

allocate(lo(nc_dom*nr_dom), stat=ios)
call check_error(ios,'Error allocating lo.',iam)

gmask = 0.0
ngeos = nc_index*nr_index
glis = nc_dom*nr_dom
ferror = 1
!-----
! Open GEOS forcing file
!-----
#if ( defined OPENDAP )
  if ( order == 1 ) then
    geos_index = get_geos_index(0)
  else ! order == 2
    geos_index = get_geos_index(3)
  endif
  write(cgeos_index, '(i4)') geos_index

  print*, 'MSG: readgeos -- Retrieving GEOS forcing file ', &
    trim(name), ' (',iam,')'
  call system("opendap_scripts/getgeos.pl "//ciam//" "// &
    trim(name)//" "// &
    cgeos_index//" "//cgeos_slat//" "//cgeos_nlat)
#endif
  print*, 'MSG: readgeos -- Reading GEOS forcing file -', &
    trim(name), ' (',iam,')'

#if ( defined OPENDAP )
  open(40,file=name,form='unformatted',iostat=ios)
#else
  open(40,file=name,form='unformatted', &
    access="direct",recl=geosdrv%ncold*geosdrv%nrold*4,iostat=ios)
#endif
  if ( ios /= 0 ) then
    print*, 'ERR: readgeos -- Error opening ', &
      trim(name), '. Stopping.', ' (', iam,')'
    call endrun
  endif

#if ( defined OPENDAP )
  do i = 1,geosdrv%nmif

```

```

        read(40,iostat=ioerror)tempvar(:, :, i)
    enddo
#else
    !read(40,iostat=ioerror)tempvar
    do i = 1,geosdrv%nmif
        read(40,rec=i,iostat=ioerror)tempvar(:, :, i)
    enddo
#endif
if ( ioerror /= 0 ) then
    print*, 'ERR: readgeos -- Error reading ', &
        trim(name), '. Stopping.', ' (', iam, ')'
    call endrun
else
    print*, 'MSG: readgeos -- Read GEOS forcing file -', &
        trim(name), ' (', iam, ')'
endif
!-----
! Finding number of forcing variables
! (13 if time step is 0, otherwise the normal 10)
!-----
if (tscount .eq. 0) then
    nforce = geosdrv%nmif
else
    nforce = 10 !change later?
endif
do v=1,nforce
!-----
! Transferring current data to 1-D array for interpolation
!-----
    c=0
    do i=1,nr_index    !i=1,lf%nroid
        do j=1,nc_index !j=1,lf%ncold
            c = c + 1
            f(c) = tempvar(j,i,v)
            if (tscount .eq. 0 .and. order .eq. 1 &
                .and. v .eq. 11) then
                gmask(c) = f(c) ! Storing geos land mask for later use
            endif
        enddo
    enddo
enddo
!-----
! Initializing input and output grid arrays
!-----
    kgdso = 0
    kgdso = ld%kgds
    if (v .eq. 8 .or. v .eq. 9) then
        ibi = 1
    else

```

```

        ibi = 1
    endif
!-----
! Defining input data bitmap
!-----
    do i=1,ngeos
        lb(i)=.true.
    enddo
!-----
! Alter default bitmap prescribed above for
! surface parameters (soil wetness, snow)
!-----
    if (v .eq. 12 .or. v .eq. 13) then
        do i=1,ngeos
            if (gmask(i)==100.0 .or. gmask(i)==101.0) then
                lb(i)=.false.
            else
                lb(i)=.true.
            endif
        enddo
    endif
!-----
! Defining output data bitmap
!-----
    do i=1,glis
        lo(i)=.true.
    enddo
!-----
! Interpolate data from GEOS grid to GLDAS grid
!-----
#if ( defined BUDGETIP )
    if(v.eq.8 .or. v.eq. 9) then
        CALL POLATES3(kgdso,ibi,lb,f,ibo,lo,go,mi, &
            rlat3,r lon3,w113,w123,w213,w223,n113,n123,n213,n223,iret)
    else
        call polates0(kgdso,ibi,lb,f,ibo,lo,go,mi,mo, &
            rlat0,r lon0,w110,w120,w210,w220,n110,n120,n210,n220,iret)
    endif
#else
    call polates0(kgdso,ibi,lb,f,ibo,lo,go,mi,mo, &
        rlat0,r lon0,w110,w120,w210,w220,n110,n120,n210,n220,iret)
#endif
!-----
! Convert data to original 3D array & a 2D array to
! fill in of missing points due to geography difference
!-----
    count = 0
    do j = 1+nroffset, nr_dom+nroffset !j = 1, ld%nr

```

```

        do i = 1, nc_dom                !i = 1, ld%nc
            if(gindex(i,j) .ne. -1) then
                tempgeos(gindex(i,j),v) = go(i+count)
            endif
        enddo
        count = count + ld%lnc
    enddo
!-----
! Fill in undefined and ocean points
!-----
    do i = 1, ld%ngrid
        if (tempgeos(i,v) >= 9.9e+14) then
            tempgeos(i,v) = ld%undef
        endif
        if(order.eq.1)then
            glbdata1(v,i)=tempgeos(i,v)
        else
            glbdata2(v,i)=tempgeos(i,v)
        endif
    enddo
enddo                !i
                    !v

print*, 'DBG: readgeos -- Deallocating arrays'

deallocate(tempvar, stat=ios)
call check_error(ios, 'Error deallocating tempvar.', iam)

deallocate(tempgeos, stat=ios)
call check_error(ios, 'Error deallocating tempgeos.', iam)

deallocate(f, stat=ios)
call check_error(ios, 'Error deallocating f.', iam)

deallocate(go, stat=ios)
call check_error(ios, 'Error deallocating go.', iam)

deallocate(gmask, stat=ios)
call check_error(ios, 'Error deallocating gmask.', iam)

deallocate(lb, stat=ios)
call check_error(ios, 'Error deallocating lb.', iam)

deallocate(lo, stat=ios)
call check_error(ios, 'Error deallocating lo.', iam)
call absoft_release_cache()

print*, 'MSG: readgeos -- Closing GEOS forcing file -', &
    trim(name), ' (' , iam, ')'

```

```
close(40, iostat=ios)
if ( ios /= 0 ) then
  print*, 'ERR: readgeos -- Error closing ', trim(name), &
    '. Stopping.', ' (' , iam, ') '
  call endrun
endif

print*, 'DBG: readgeos -- leaving', ' (' , iam, ') '

return
```

1.1.1 check_error (Source File: readgeos.F90)

Error check; Program exits in case of error

INTERFACE:

```
subroutine check_error(ierr,msg,iam)
```

CONTENTS:

```
if ( ierr /= 0 ) then
  print*, 'ERR: readgeos -- ',msg, ' Stopping.', ' (' ,iam, ') '
  call endrun
endif
```